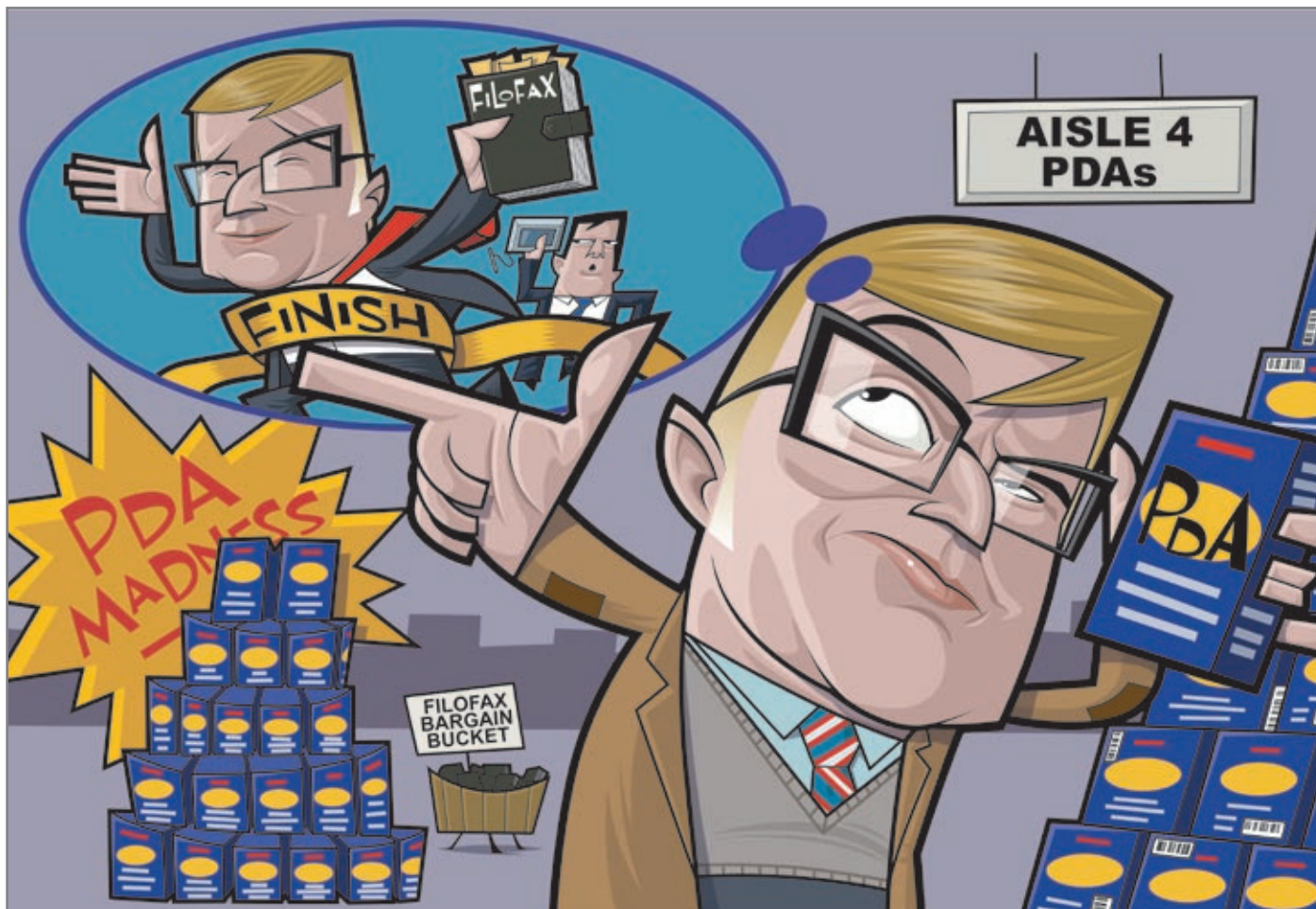




UNDER DEVELOPMENT

In his quest to find a portable solution for barcode readers, David Robinson finally succumbs to the lure of a personal digital assistant. And then he gets another one, just for good measure

I'm feeling quite sorry for myself as I write this, because I have a humdinger of a cold (currently pronounced 'code'), which I caught off programmer Blakey. He's always ill with something or other; he blames this on having two small daughters who, he says, get infected at school then pass it on to him. Mind you, I wonder if Blakey's fit to be a dad when he tells his little girls that ice cream vans only make 'that noise' when they've run out of stock.

In between the sniffles, I've done something I once swore I'd never do, as well as learned a new programming

language. The something I swore I'd never do is buy a PDA. Long-term readers will remember my feeble attempts at getting one to do handwriting recognition last year and, even further back, the 'find-the-phone-number' races with a friend – me with my Filofax and him with his PDA. I always won.

RAISING THE BAR

Still, needs must. I've spent a year, on and off, looking for the perfect portable data collection setup that can be used with a variety of barcode readers. On a couple of occasions I've

come quite close to the perfect solution, only to fall at some final stumbling block.

The problems we need to overcome are as follows:

- The barcodes are often either high up or in a fairly inaccessible place
- The people doing the scanning aren't always very diligent, so you have to have a simple user interface that allows them to do only what you want them to do
- You have to be able to do these things anywhere, anytime, not just in a local environment (such as a

warehouse) that's set up for RF scanners. Besides, that would be far too expensive for anything the size of a warehouse

- The capturing device sometimes needs to be fed with data that has to be verified against what's being scanned, so you can stop people dispatching the wrong thing to the wrong place
- Users have to be prevented from capturing spurious data by, say, seeing what happens if you scan a KitKat
- You have to be able to invoke different kinds of checking logic because what you want to scan and check varies according to what your customer's customer (aka cus2mer) wants and the data he provides you.

This is a tough list. All the data-handling requirements can be done with a small portable PC, and I've devised a command system totally based on barcodes that eliminates the need to input anything using either a keyboard or mouse. Even better, with this system you don't even need to look at the screen to get feedback or instructions on what to do next. Voice synthesis provides all the responses, and with careful design it's impossible for the user to input anything wrong or out of sequence. This all runs perfectly in test situations



on a little computer such as the JVC sub-notebook I tried last year. Unfortunately, being a PC, there are just too many ways that users Blakey would call muppets can mess up, and the package wasn't compact or robust enough to deploy in a workplace.

I almost thought we'd cracked it when I discovered what must be the world's tiniest proper PC, a thing called a SaintSong Espresso. If you do a Google search for 'saintsong espresso' with the 'pages from the UK' option ticked then you should get a page at www.computerbargains.co.uk that gives full details of this tiny marvel. It's about two-thirds of the size of a VHS tape and you can set it up to run without a keyboard or screen, so it's ideal for my barcode/voice interface. If the application is set to autorun on startup, there are very few ways for the muppets to screw up.

Only two things stopped this being absolutely perfect: (a) to add a Bluetooth interface, you had to use an external 'dongle' type that stuck out of the side of the Espresso and was a bit vulnerable; and (b) the power requirements – it needed 15 volts DC, which is a tricky rating to supply from batteries. The nearest I could get to a decent portable power supply was two 7.2 volt lithium-ion camcorder batteries wired in series. This did the job and would power the Espresso for over two hours, but it was inelegant and had to be dismantled to charge the batteries.

LOOX PROMISING

We had a great concept for solving the problem but had to go back to the drawing board. Then I saw an ad for a new PDA called the Loox from Fujitsu. There are two major derivatives: the 400 series and the 700 series. The 700 has both Bluetooth and regular WiFi built in. Out of desperation, I felt compelled to give one a try. I wanted a 720, which has a built-in camera, but the distributors were out of stock so I settled for a cameraless 710 that was a little cheaper at around £340.

The Loox runs Microsoft Pocket PC, which is sort of familiar. However, what I knew about programming these things could be written on a small postage stamp. Time for some research. Searching the web, I became almost more confused than when I started, so I decided to buy a book. There's very little to choose from. I settled on *Palm and Pocket PC Programming* by Vadim Kalinin and Vladimir Raflovich, largely because it seemed to offer coverage of several language options, as well as both major PDA platforms. The 'obvious' language to use would be Embedded

BOOK OF THE MONTH

Palm and Pocket PC Programming

RATING ★★ ★

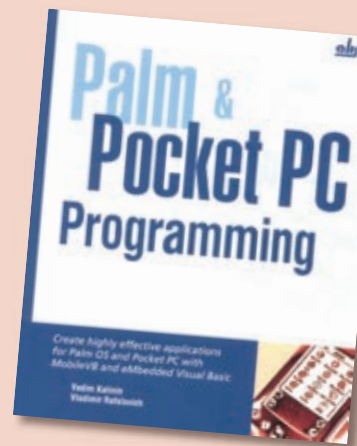
PRICE £20

SUPPLIER www.amazon.co.uk

ISBN 1931769206

PAGES 330

This book is not perfect; it covers only App Forge and EVB examples and is also a bit out of date, having been published in 2003. But there's not much else available if you want to learn PDA programming.



VB (EVB) from Microsoft but, apart from my built-in allergy to Microsoft languages, I was put off by the sheer size of the download: over 250MB in compressed format.

Kalinin and Raflovich list most of the development languages for PDA-type computers. Some work only with Palm or Pocket PC, while some do both. After discussing some of the relative merits of each language (albeit very briefly in some instances), the book majors on App Forge Mobile VB and the Microsoft Embedded VB for all its examples. App Forge looked OK but, before shelling out for a copy, I took a look at a language called

interpreted when you run the program using the run-time module that you also have to install (just once) on your PDA. Using this method, CASL programs are usually very small and, despite the interpretation process, run quickly.

The CASL language looks to me like a hybrid, with Pascal, Basic and C characteristics. In the main, though, it's easy to learn and comprehensive. It can create and manipulate files and talk to serial ports and network connections. A few experiments confirmed that I could build a program to do most of what I wanted, so I risked the princely sum of \$84

Long-term readers will remember my 'find-the-phone-number' races with a friend – me with my Filofax and him with his PDA. I always won

Compact Application Solution Language (CASL, www.caslsoft.com). What attracted me were the words "resembles Pascal in its syntax". Kalinin and Raflovich gave it no more than a six-line paragraph, but I'm glad I took the time to research it.

CODE COMFORT

The CASL people are nice enough to give you a usable demo version that you can download. Unlike Visual Basic, this is a mere 5.5MB and includes all the help documentation. The demo is limited to 40 global variables, 20 functions and 20 objects. Otherwise, it works like the normal version. The CASL environment allows you to develop and debug your programs on your PC. When you're happy with the result, you set the compiler to produce either a Palm or Pocket PC program, recompile your code then hot sync the result to your PDA.

By default, CASL produces an intermediate program that is

(around £46) for a proper copy. There are a couple of add-on options that get you a compiler, which produces standalone executables; for Pocket PC this costs \$50 (around £27) so your programs don't need the run-time module and run even faster than the interpreted ones.

PORT IN A STORM

Then I hit a snag. CASL addresses COM ports 1 to 4 on a Pocket PC device. When I 'paired' our Bluetooth barcode scanner to the Loox, it insisted on taking COM 8, and there's no way of shifting it – damn. Worse, Keef downloaded the monster EVB stuff and modded one of the demo programs to talk to the scanner. Could we read data from the port? Nope. Even worse, I found a program called CEWedge from www.taltech.com, which claims to interface Pocket PCs to Bluetooth scanners and couldn't get that to work either. Is there something 'odd' about the way the Loox implements Bluetooth? Dunno.

What's more, I don't know how to find out either. Double damn.

PULL THE OTHER ONE

In my endless net browsing for a solution, I discovered a CASL add-on library from an independent outfit called Brainyware run by a guy called Paul Steinmeyer. This stuff runs only with the Palm version of CASL and, of course, programs using it run only on Palm PDAs. What the hell. At least with my CASL program I wouldn't have to rewrite it. I ordered a Palm Zire 72 with built-in Bluetooth (and a camera) and the CASL Pro extended compiler. I actually like it better than the Loox, and it's half the price. Add in Steinmeyer's library code (by the way, his tech support is excellent) and interface it to my application... does it talk to my scanner? It does. Oh yes! I'm so pleased, I do a reasonable impression of Meg Ryan in the famous diner scene in *When Harry Met Sally*.

Now all I had to do was build a method for getting data in and out of the PDA in a seamless fashion. I was ready to do this using a variation of the regular Hotsync between PDA and PC with a program running as a service on the PC to identify new incoming stuff, process it and send it to the customer's mission-critical program.

However, Keef had also been experimenting and came up with something more elegant. He built a variation on an SMTP mail server that can run on any networked PC or server. It 'understands' commands sent to it by other programs and can send and receive data. Using CASL's built-in network interfacing, we added a feature to my program that 'talks' to Keef's server program. Now I could transmit data both ways directly from my Palm to the main database.

I showed a working example to several clients and they've all been impressed, so now we're moving from the prototype to the working program stage.

WINDS OF CHANGE

While all this was going on, I had to deal with a demand from Keef to move Blakey out of the developer's office. He's always been 'windy' (if you get my drift), but he read last week in a newspaper that farting makes you live longer. He's taken to periodically exclaiming, "another 15 minutes," upon which everybody gets up to make coffee. I think I know where to put his desk – in the car park. ☹

CONTACT

DAVID ROBINSON

davidr@computershopper.co.uk